J. Software Engineering & Applications, **2009, 2:60-65**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

FEL-H Robust Control Real-Time Scheduling

Bing Du[1], Chun Ruan[2]

[1]School of Electrical and Information Engineering, University of Sydney, Australia, [2]School of Computing and Mathematics, University of Western Sydney, Australia
Email: bing@ee.usyd.edu.au, c.ruan@uws.edu.au

Received January 26th, 2009; revised February 28th, 2009; accepted March 25th, 2009.

ABSTRACT

The existing scheduling algorithms cannot adequately support modern embedded real-time applications. An important challenge for future research is how to model and introduce control mechanisms to real-time systems to improve real-time performance, and to allow the system to adapt to changes in the environment, the workload, or to changes in the system architecture due to failures. In this paper, we pursue this goal by formulating and simulating new real-time scheduling models that enable us to easily analyse feedback scheduling with various constraints, overload and disturbance, and by designing a robust, adaptive scheduler that responds gracefully to overload with robust H∞ and feedback error learning control.

Keywords: *Robust*, *Real-Time Scheduling*, *Feedbacke*, *Simulation*, *Feedback*

1. Introduction

Feedback control is a powerful tool to make real-time scheduling robust towards external and internal disturbances and uncertainties. Feedback techniques were originally proposed in time sharing systems and have been successively applied to real-time and multimedia systems. Seto *et al*. [1] proposed integrating computer-control and real-time system design so that the performance of a task is a function of its sampling frequency, and identified an optimization problem to find a set of optimal task periods. Lu *et al*. [2] proposed a feedback scheduler based on earliest-deadline first scheduling (EDF), PID controller, and a more theoretically founded approach. Using a PID controller is often apposite in many industrial applications and in feedback control scheduling, but robustness is not guaranteed. Papers [3] integrated feedback control with model-based prediction to anticipate and correct future delay fluctuation. [4] proposed measuring, quantifying, adapting and bounding miss ratio and average system utilisation. These techniques addressed feedback priority-based scheduling literature to ensure that no deadlines are missed. Cervin *et al*. [5] combined feedback with feedforward to allow the scheduler to compensate for resource changing before any overload occurred. Their techniques are tailored to computing systems that may be modelled as sets of digital control loops.

However, no theoretical analysis has been provided about how to model a real-time computing system that includes the effects of the sampling rate, the jitter, actuation, plant uncertainty and nonlinearity, and how to design a robust real-time controller to optimise soft real-time system performance. The main obstacle that prevents control theories from being applied effectively in computing systems is how to construct a computing model that works in open and unpredictable environments. On the other hand, existing feedback scheduling approaches find it difficult to guarantee robust performance properties, as they only use a simplistic maximum constant model and "classical" proportional-integral-derivative PID to design a complex real-time scheduling system. In many cases, the PID controller design techniques are a satisfactory solution. It seems unnecessary to apply more powerful tools. However, when the scheduling system dynamics are complex and poorly modeled, or when the performance specifications are particularly stringent, no solution is forthcoming.

This paper aims at introducing advanced modern control theory to analyze and design real-time systems. The goal of our research is to investigate a real-time scheduling model that can be applied easily to different real-time systems, and proposes a new control scheduling algorithm that is more effective than PID feedback scheduling. We present a two-tank system that is used to simulate a dynamic real-time scheduling model and FEL-H (Feedback error learning and H∞ control) scheduling. The main contributions of this paper are as follows:

1) We use modern control theory as a theoretical foundation to analyze and design adaptive real-time scheduling. In contrast with most of the existing feedback scheduling algorithms that use an ad-hoc manner or PID algorithms, we employ H∞ and FEL control theory as a rigorous methodology to achieve faster response speed and more robust performance guarantees in unpredictable environments.

2) Traditional real-time scheduling theories depend on accurate a priori knowledge of the system workload pa-

rameters. Existing feedback scheduling algorithms cannot respond quickly to workload or model changes and guarantee robust system performance. Our scheduling algorithm, based on feedback error learning control, always tracks the scheduling system accurately and quickly. This feature is especially valuable for performance- critical systems such as on online trading, stock, e-business servers and defense applications. We also consider how to obtain an optimal sampling period and compensate for jitter that is usually not taken into account in existing feedback scheduling. Our H feedback control design methodology provides robust and analytical performance guarantees for open systems, despite workload uncertainties.

3) Unlike traditional physical electrical and mechanical control systems that have a finite set of ordinary differential equations as a design model, the dynamics of real-time computing systems are too complicated to capture the essential features of the scheduling systems. We propose new H∞-norm performance indexes that include the effects of inputs and disturbances on error and control signals. A two-tank system is analysed to simulate a dynamic scheduling model. This model enables us to model and analyse feedback scheduling with various constraints, overload and disturbance more exactly and easily. Furthermore, our new robust FEL-H feedback scheduling integrates H∞ robust optimal control theory, feedback error learning control theory and scheduling theories that do not require precise system model parameters.

4) The existing simulators are available only for a few schedulers and task models, and do not support new scheduling policies, such as H∞-norm feedback scheduling. Our feedback control scheduling simulator (FCSS) allows us to explore various approaches of feedback control real-time scheduling and constraints.

The rest of the paper is organised as follows. We present FEL-H scheduling architecture in Section 2. Section 3 gives how to model a FEL-H real-time scheduling system. In Section 4, we show the robust FEL-H feedback scheduling design. Experiment is discussed in Section 5. The paper is concluded, and suggestions for future work are presented in Section 6.

2. FEL-H Scheduling Architecture

To apply control theory methodology to scheduling, the controller should not only stabilize the nominal real-time kernel, but also meet performance specifications for all possible real-time kernels defined by the uncertainty. A typical FEL-H control scheduling architecture is composed of a feedforward controller Q, feedback controller K, task actuator, QoS actuator, EDF scheduler and CPU (as illustrated in Figure 1). The CPU, EDF scheduler, QoS actuator and task actuator can be treated as a basic scheduling model P. The objective of the control is to minimise the errors between the reference utilisation, deadline miss ratio, and system output utilisation U and deadline miss ratio M.

The objective of the control is to minimise the errors between the reference utilisation, deadline miss ratio, and system output utilisation and deadline miss ratio. An H ∞ controller attempts to keep the CPU utilization U at a high level, avoid overload, distribute the computing resources, and maintain the number of missed deadlines as low as possible. It computes the amount of CPU load that is added into or reduced from the system. An FEL controller will respond rapidly to nonlinear saturation, keep errors near 0 and pull U to return to a linear range. The task actuator controls the amount of workload into the system, and the QoS actuator adjusts the workload inside the system, so that the system accepts as many tasks as possible while minimising the deadline-miss ratio of all the submitted tasks.

The sensor monitors and measures the controlled variables and sends the M(k) and U(k) back to the controller.

The feedforward controller, which contains tunable parameters, controls the utilisation to respond quickly to workload changes. The H∞ feedback controller enables the scheduling system to guarantee robust system performance and to compensate for model error. The output of the H∞ controller regulates and trains the inverse scheduling dynamics model on-line. Therefore, the FEL-H scheduling controller captures, trains and controls at the same time. The EDF scheduler schedules the accepted tasks according to the EDF policy, which can achieve a deadline miss ratio of 0% if requested utilisation is less than 100%. The FEL-H controller is a control computation algorithm to be executed in every sampling period h. A set of tasks will share the CPU. These tasks are control tasks or scheduling tasks. They perform sampling, control computation and actuation. The deadline-miss ratio and CPU utilisation reference are Mr and Ur. The robust requirement is introduced into the design by imposing individual weighting functions on the uncertainties, external disturbances, and the performance specifications.

The Task actuator decides which workload will be allowed into the system and which will be denied. Whether the system should admit or reject requests depends on the task's request utilisation. If the requested utilisation of the incoming task results in a total CPU requirement of all tasks less than Ur = 90%, the task will be admitted, otherwise it will be rejected.

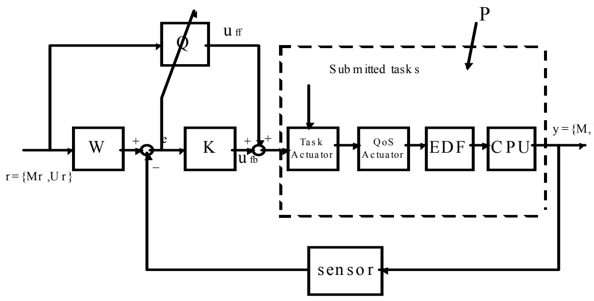

Figure 1. Architecture of FEL-H control scheduling

The task actuator may further adjust to admit more workload if the QoS actuator degrades the QoS level. It can also reject more workload if the QoS actuator upgrades the QoS level.

In the QoS control scheme, each task has different resource requirements for each discrete quality level. The system maintains a single value that represents the quality level of the overall system and is called the "QoS level". The QoS level determines how to allocate resources to each task. If the QoS level downgrades, the resource allocations of some tasks are decreased. Although many QoS control policies are proposed, they are not suited for real-time systems that need to keep the timing constraints. We use a table containing the resource requirements of all tasks [8]. Resource allocations for tasks and total resource utilisation can be obtained from the table. The QoS table allows the system designer to specify a QoS control policy. The QoS actuator changes the requested utilisation in the system by adjusting the service levels of accepted tasks. It will return the portion of tasks not accommodated to the task actuator.

The FEL-H architecture can use different real-time scheduling policies (such as EDF or Rate/Deadline Monotonic) as basic schedulers to schedule admitted tasks. There are significant different performance references for different basic scheduler policies when we design the FEL-H scheduling system.

3. Modeling a FEL-H Real-Time Scheduling System

To implement robust scheduling, our research [6,7,9] proposes a two-tank system model to emulate a scheduling system (Figure 2). The progress of a task request queue is similar to a fluid flow into a multi-tank system. They have the same dynamics due to their intrinsic queuing structure. The system output is the utilisation that is mapped to the liquid level h in tank 2 and input is represented by admitted task R that is mapped to the flow u into tank 1.

The tasks accepted by the CPU are simulated as liquid flowing into the CPU, and the tasks completed by the CPU are viewed as liquid flowing out of the CPU. The CPU can be viewed as a liquid tank that inputs liquid (accepting tasks) and outputs liquid (completing tasks). The tasks submitted to a real-time scheduling system are viewed as liquid that wants to flow into a real-time scheduling system. They may not be equal to the tasks accepted by the CPU. The task actuator decides whether to accept or reject submitted tasks. Therefore, submitted tasks cannot flow directly into the CPU tank. It is impossible to represent tasks accepted by the task actuator and tasks accepted by the CUP if we only use one tank. The tasks that are accepted by the task actuator are simulated as liquid flowing into the real-time scheduling system. The QoS actuator will decide whether this liquid can flow into the CPU tank or not. The part of the liquid that flows into the CPU tank represents the tasks accepted by the CPU. The other part, which does not flow into the CPU

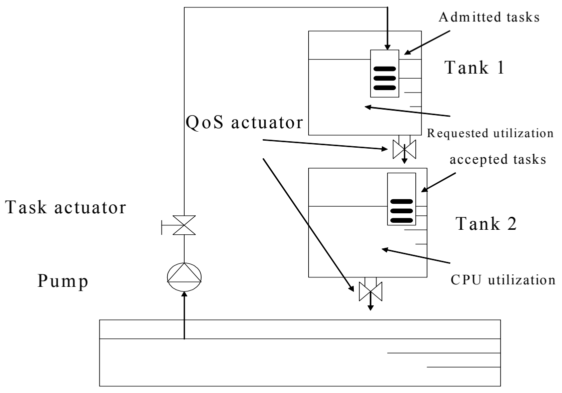

Figure 2. Two-tank system model of a scheduling system

tank, will stay in the real-time scheduling system. This part of the real-time scheduling system can be simulated as another tank. The heights of the liquid levels of the two tanks are coupled together and interact. They are a complex nonlinear, time-varying and multivariable system that is an approximate abstraction of the real-time scheduling system. The two-tank system model is sufficiently accurate to simulate and analyse a real-time scheduling system, as our experiments will show.

Level 2 and level 1 in tank2 and tank1 represent requested utilization and CPU utilization as shown in Figure 2. Our goal is to design a controller that regulates the CPU utilisation, keeping it at 90%. This is mapped to design a controller so that the level in tank 2 is regulated to the reference value by the task actuator and the QoS actuator. The transfer function of the scheduling system can be written as follows:

$$\frac{U(s)}{R(s)} = \frac{K^*}{(T_1 s + 1)(T_2 s + 1)} \tag{1}$$

where:

$$K^* = \frac{1}{\Phi}\sqrt{\frac{2U_0}{g}}$$

The time constants T_1 and T_2 are related to performance reference points and the QoS actuator that are mapped to the level in the tanks, Φ the outlet's cross-sectional area and the cross-sectional area of the tanks.

4. Design of FEL-H Real-Time Scheduling

We introduce the usual form and generic H∞ block diagram to represent the scheduling systems shown in Figure 3. The FEL-H controller should make the system stable, and satisfy the steady state and transient state performance specifications. The system output comprises the controlled variable miss ratio M(k) and utilisation U(k). The input signals to the scheduling system include the performance reference and disturbance input. The per-

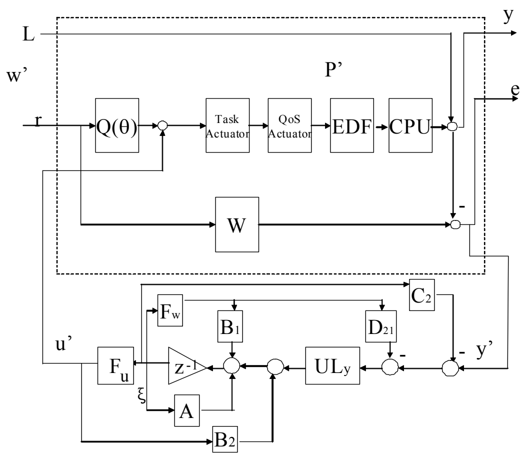

Figure 3. FEL−H Scheduling Controller

formance references Mr and Ur use a step signal. The internal overload that adds the total requested utilisation by the admitted tasks' CPU utilisation variation is modelled as a disturbance L. A step load is used as disturbance because it represents severe load variations. The stabilising H∞ controller minimises the transfer function matrix mapping to. This ensures scheduling-system performance-tracking and workload disturbance attenuation by keeping small.

The H∞ controller of scheduling systems can be described as the central controller (2) (3) (4) (as shown in Figure 3):

$$\xi(k+1)=A\xi(k)+B_1\eta(k)+B_2u(k)-UL_y(y(k)-C_2\xi(k)-D_{21}\eta(k)) \quad (2)$$

$$u(k)=F_u\,\xi(k) \quad (3)$$

$$\eta(k)=F_w\,\xi(k) \quad (4)$$

The FEL controller Q will further minimize as (5).

$$Q(z)=\frac{[4k_w+2T(f_2k_w+c_{w,2})+T^2(f_1k_w+c_{w,1})]z^2+2[T^2(f_1k_w+c_{w,1})-4k_w]z+[4k_w-2T[f_2k_w+c_{w,2}]+T^2(f_1k_w+c_{w,1})]}{[4+2T(f_2-d_{w,2})+T^2(f_1-d_{w,1})]z^2+2[T^2(f_1-d_{w,1})-4]z+[4-2T(f_2-d_{w,2})+T^2(f_1-d_{w,1})]} \quad (5)$$

5. Experiment

We developed a simulation environment to evaluate the performance of our FEL-H scheduling algorithms. The controllers can be designed by using MATLAB tools. Our robust FEL-H scheduling will still achieve performance specifications even with uncertain model parameters and unpredictable operating environments. This lets our FEL-H scheduling be directly applied to different systems and it does not need re-design for every system. We can compute the prefilter, feedback error learning controller and H∞ controller parameters. The sampling period T is 0.5 sec. The results of the FEL controller parameters are listed in Table 1.

The H∞ controller can be derived.

$$\xi(k+1)=\begin{bmatrix}3.45 & 4.71\\ 1.84 & 2.39\end{bmatrix}\xi(k)+\begin{bmatrix}2.34\\ 1.72\end{bmatrix}\eta(k) \quad (4)$$

$$u(k)=\begin{bmatrix}-2.31 & 0.59\end{bmatrix}\xi(k)-1.73\,\eta(k) \quad (5)$$

$$\eta(k)=y(k)-\begin{bmatrix}7.85 & 2.36\end{bmatrix}\xi(k) \quad (6)$$

The utilisation reference should be less than the nominal threshold of the EDF scheduling policy, so that the utilisation error will not remain at 0 when the system is overloaded. Otherwise, the system will stay in overload. The theoretical bound is 100% for EDF and the periodic task set. We set the utilisation reference Ur=90%. The miss ratio reference will change, since different ap- plications have different requirements and tolerances to missed deadlines. For example, the stock-trading transactions have more strict timing constraints than usual online trading. The miss ratio reference is chosen as Mr=2.5% in our experiment.

Table 1. FEL Controller Parameters

w_1	w_2	w_3	k_w	f_1	f_2	c_1	c_2	d_1	d_2
8	16	1	18.43	5.26	5.75	3.59	2.41	7.37	9.82

A set of tasks arrives suddenly with a total CPU utilisation of 150% at the highest QoS level. The workload increases from zero to 150% overload. All experiment algorithms use the same EDF scheduling policies and table-based QoS optimisation algorithm. Every simulation experiment is repeated 22 times to ensure the evaluation is accurate. We describe the experiment's results for EDF scheduling, PID feedback scheduling, H∞ control scheduling, FEL scheduling and FEL-H scheduling algorithms in response to a new arrival 150% overload, and compare the results. Then, we present the performance evaluation of these algorithms in response to the disturbance of the system's internal parameters.

In this section, we present the five different algorithms in response to a 150% overload (as shown in Figure 4).

EDF scheduling has a miss ratio that is too high and fails to provide performance guarantees. The PID controller cannot provide satisfying robust performance guarantees and because the overshoot of miss ratio and average miss ratio are too high, and the durations of the settling time and rise time of U(k) are too long, H∞ feedback can provide very robust performance for system uncertainty, but the response time is too slow. FEL scheduling has the fastest rise time, but the overshoot and average miss ratio are higher than for PID and H∞ scheduling. It cannot provide robust performance guarantees for model uncertainty. FEL-H scheduling can provide the desired robust performance guarantees that consist of fast rise time and settling time, low average miss ratio and high CPU utilisation, as it can obtain a fast response from the FEL controller and more robustness from the H∞ controller.

Five different algorithms are in response to that the model parameters of the scheduling system are changed by 30%, while the new overload is 150%. (as shown in Figure 5). This case can be used to simulate the robustness of the feedback scheduling algorithms, and to demonstrate whether our scheduling algorithms can be applied to different real-time scheduling applications without needing to change the parameters of the controllers. Only the FEL-H scheduling can remain in the steady state while the utilisation U(k) remains close to 90%, and the miss ratio M(k) remains at 0 through running.

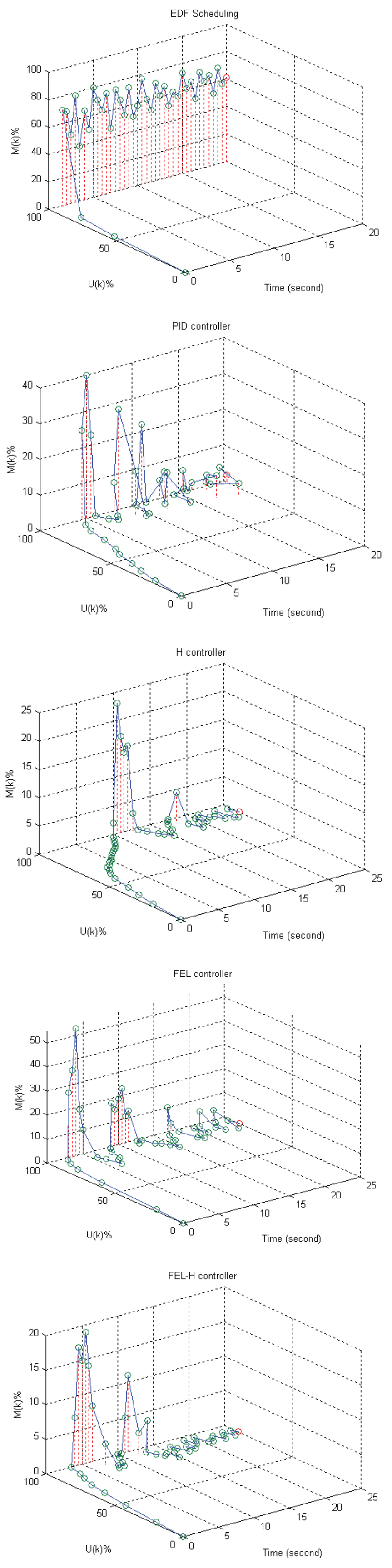

Figure 4. EDF, PID, H∞, FEL and FEL-H scheduling

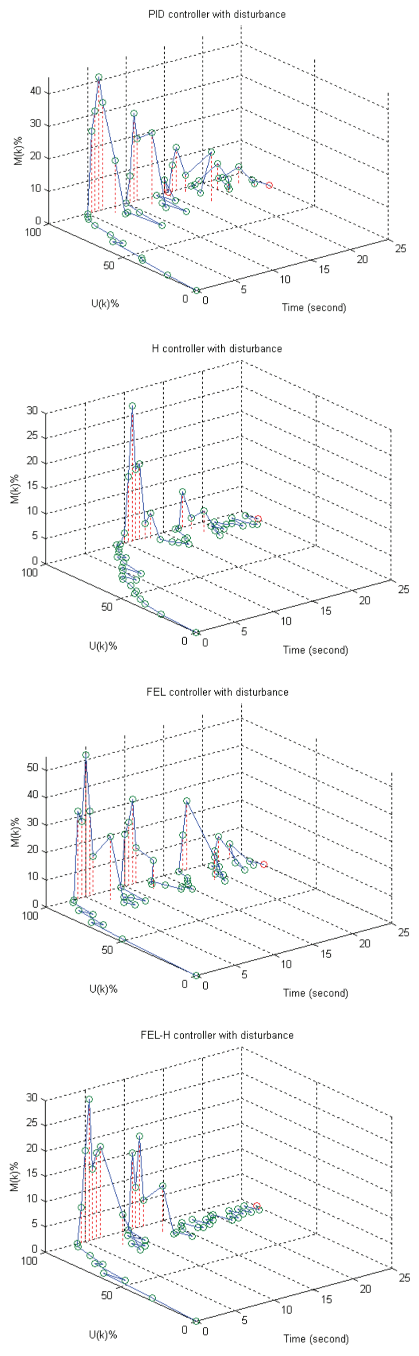

Figure 5. EDF, PID, H∞, FEL and FEL-H scheduling with disturbance

6. Conclusions

In this paper, we integrates H∞ control theory and feedback error learning control theory to model and deal with feedback real-time scheduling with uncertainty and nonlinearity. Our mechanism provides a systematic and theoretic platform for investigating how to deal with uncertainty and additive disturbances for real-time systems. The FEL-H scheduling algorithms can provide robust stability, low deadline miss ratio for real-time system uncertainty parameters and disturbance when the workload changes dramatically. Further work will improve the feedback scheduling scheme so that heterogeneous modelling and design of real-time and embedded system can be integrated. We will apply our FEL-H control scheduler to a realistic real-time system.

REFERENCES

[1] D. Seto, J. P. Lehoczky, L. Sha, and K. F. Shin, "On task schedulability in real-time control systems," In Proceedings 17th IEEE Real-Time Systems Symposium, Washington, DC, pp. 13–21, 1996.

[2] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," In Proceedings 21st Systems Symposium, Orlando, Florida, pp. 13–23. 2000.

[3] D. Henriksson, Y. Lu, and T. Abdelzaher, "Improved prediction for Web server delay control," In Proceedings of the Euromicro Conference on Real-Time Systems, Catania, Sicily, pp. 61–68, Italy, 2004.

[4] T. Abdelzaher, V. Sharma, and C. Lu, "A utilization bound for aperiodic tasks and priority driven scheduling," IEEE Transactions on Computers 53(3): pp. 334–350, 2004.

[5] A. Cervin, J. Eker, B. Bernhardsson, and K. E. Årzén, "Feedback-feedforward scheduling of control tasks," Real-Time Systems Vol. 23, No. 1, pp. 113–120, 2002.

[6] B. Du, and D. Levy, "A robust control real-time scheduling design," special issue: Advanced Control and Real-Time Systems of DCEMP journal, Vol. 13, No. 3–No.4, pp. 335–340, 2005.

[7] B. Du and D. Levy, "H∞ robust scheduling design methodology in real-time systems," In the Proceedings of the 2003 International Conferenceon Embedded Systems and Applications, pp. 285–291, USA, 2003.

[8] S. Tomoyoshi and T. Kosuke, "Table-based QoS control for embedded real-time systems," ACM SIGPLAN Notices Vol. 34, No. 7, pp. 65–72, July 1999.

[9] B. Du and C. Ruan, "Robust feedback scheduling in distributed embedded real-time systems," In the Proceedings of the 6th IEEE/IFIP International Conference On Embedded and Ubiquitous Computing (EUC 2008), pp. 90–96, Shanghai, China, 2008.